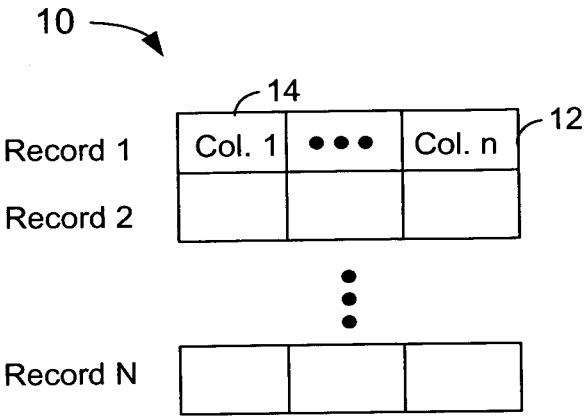

Prior Art

Fig. 1

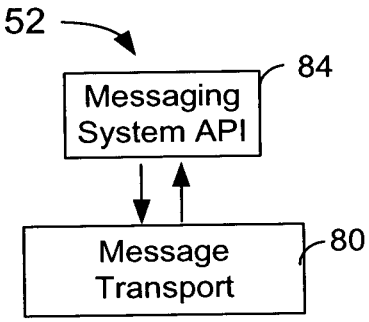

Fig. 4

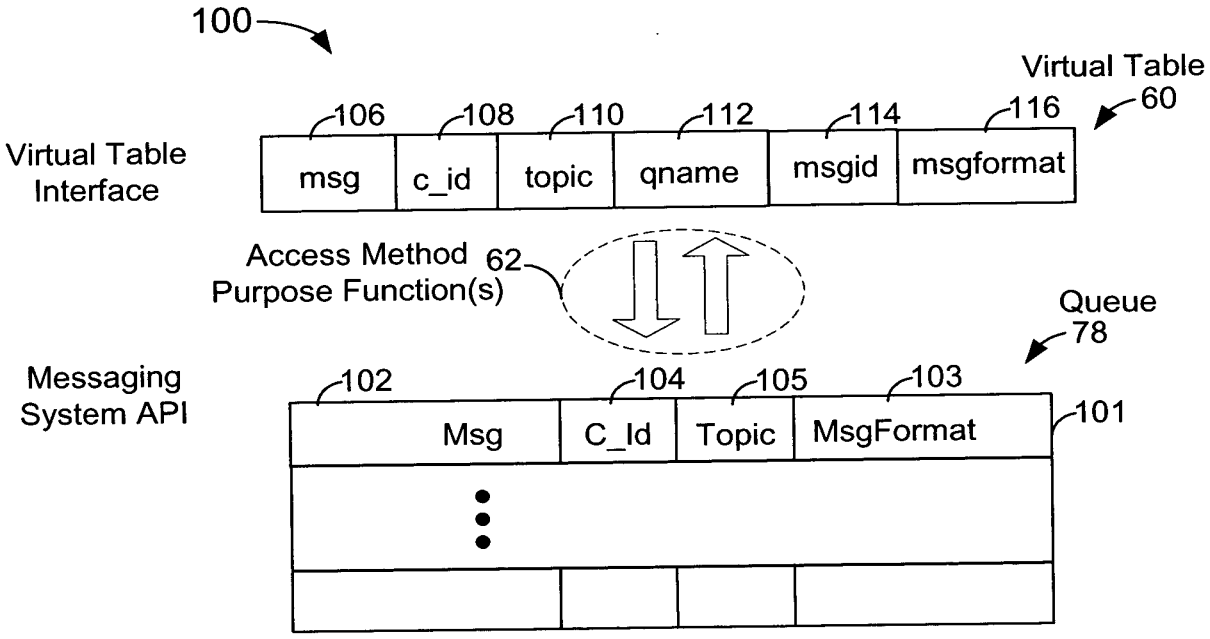

Fig. 3

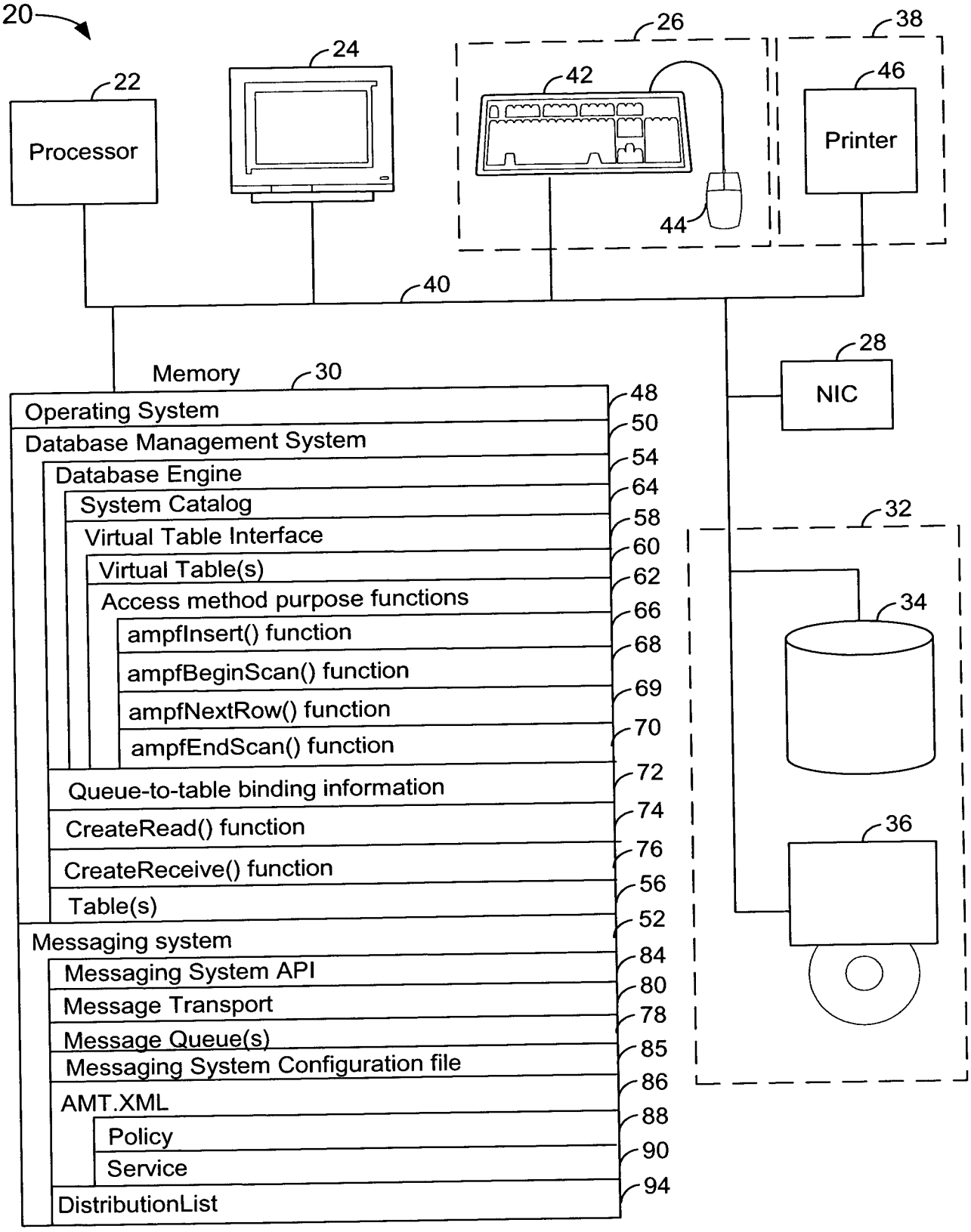
20
22
Processor
24
26
42
44
38
46
Printer
40
28
NIC
Memory
30
Operating System
48
Database Management System
50
Database Engine
54
System Catalog
64
Virtual Table Interface
58
Virtual Table(s)
60
Access method purpose functions
62
ampfInsert() function
66
ampfBeginScan() function
68
ampfNextRow() function
69
ampfEndScan() function
70
Queue-to-table binding information
72
CreateRead() function
74
CreateReceive() function
76
Table(s)
56
Messaging system
52
Messaging System API
84
Message Transport
80
Message Queue(s)
78
Messaging System Configuration file
85
AMT.XML
86
Policy
88
Service
90
DistributionList
94
32
34
36

Fig. 2

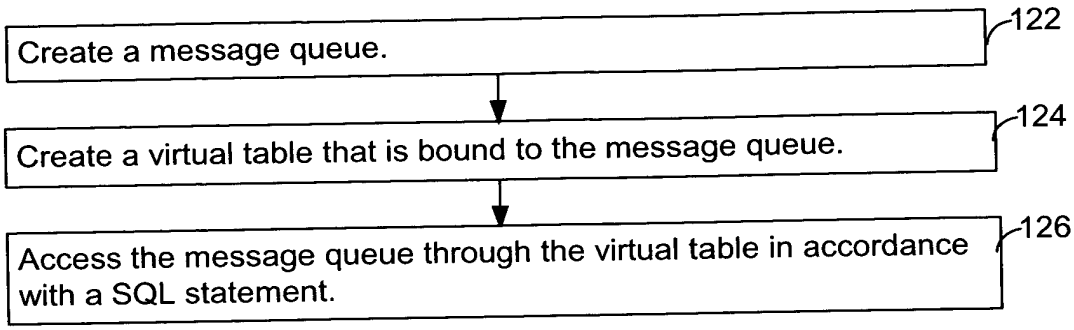
Create a message queue.
122
Create a virtual table that is bound to the message queue.
124
Access the message queue through the virtual table in accordance with a SQL statement.
126

Fig. 5

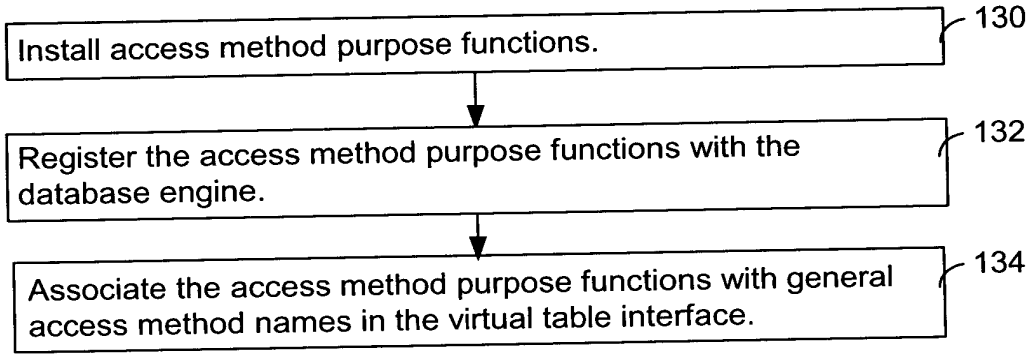
Install access method purpose functions.
130
Register the access method purpose functions with the database engine.
132
Associate the access method purpose functions with general access method names in the virtual table interface.
134

Fig. 6A

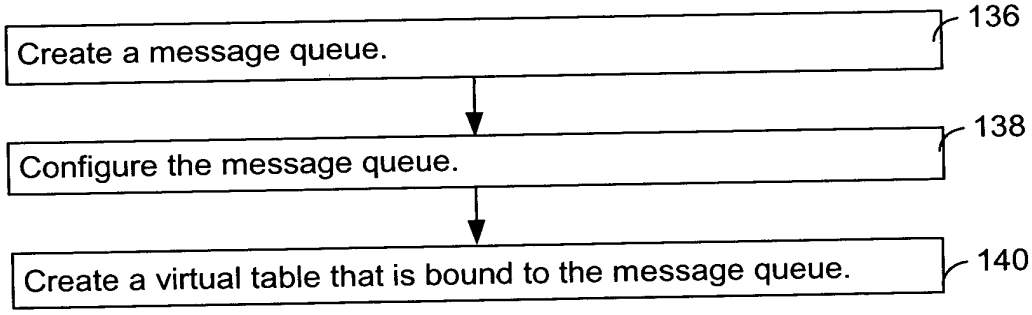

Fig. 6B

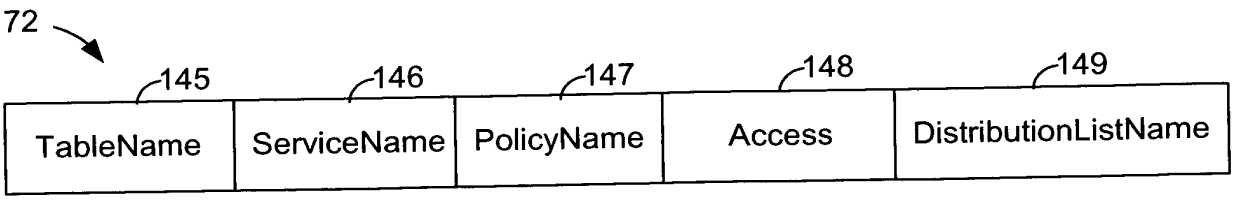

Fig. 7

150

INSERT into TABLENAME (msg, c_id) values ('message string', 'COID')

Fig. 8

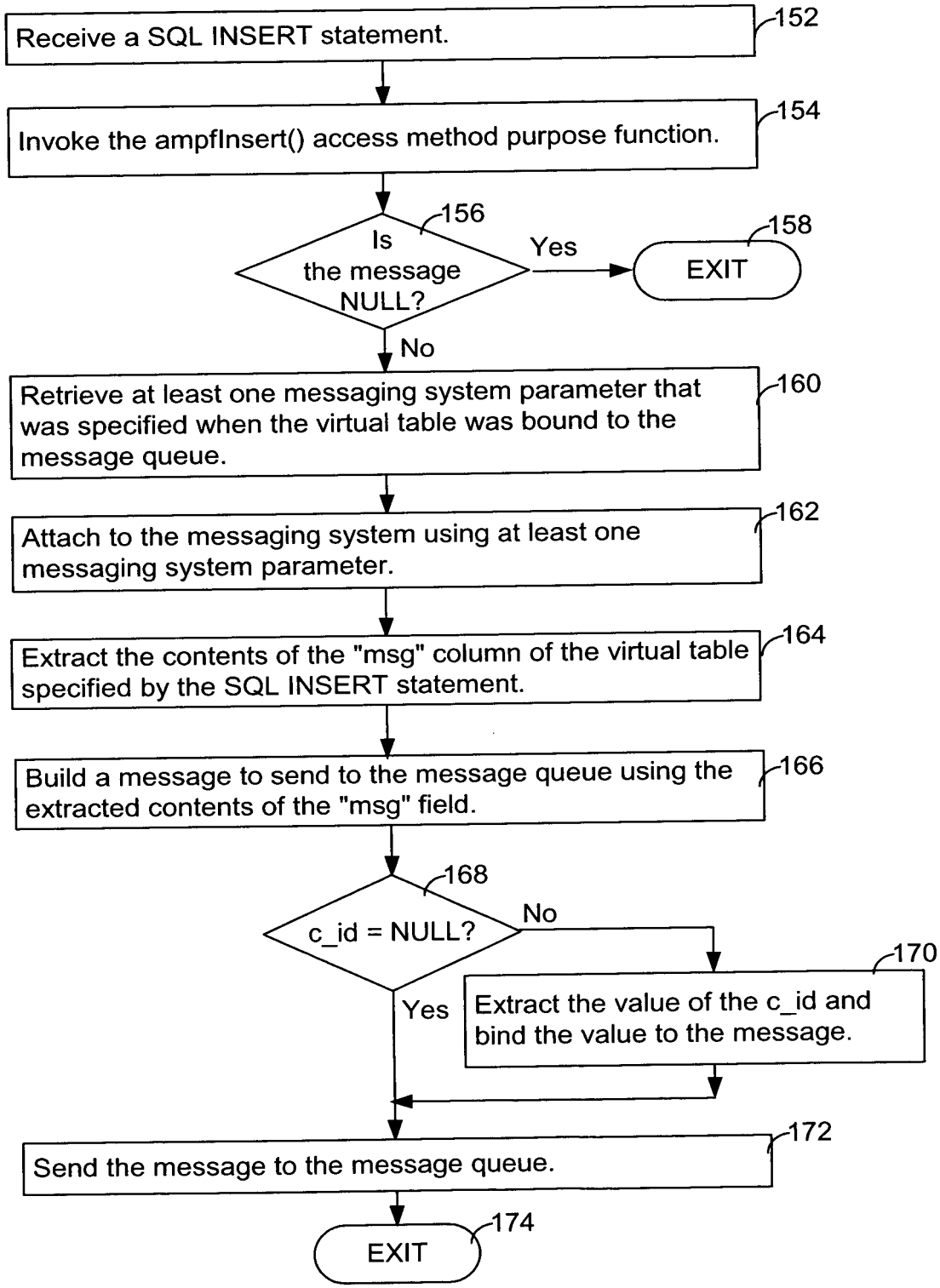
Receive a SQL INSERT statement.
152
Invoke the ampfInsert() access method purpose function.
154
156
Is the message NULL?
Yes
158
EXIT
No
Retrieve at least one messaging system parameter that was specified when the virtual table was bound to the message queue.
160
Attach to the messaging system using at least one messaging system parameter.
162
Extract the contents of the "msg" column of the virtual table specified by the SQL INSERT statement.
164
Build a message to send to the message queue using the extracted contents of the "msg" field.
166
168
c_id = NULL?
No
170
Yes
Extract the value of the c_id and bind the value to the message.
Send the message to the message queue.
172
EXIT
174

Fig. 9

180

```
SELECT msg FROM TABLENAME WHERE c_id = 'COID'
```

Fig. 10

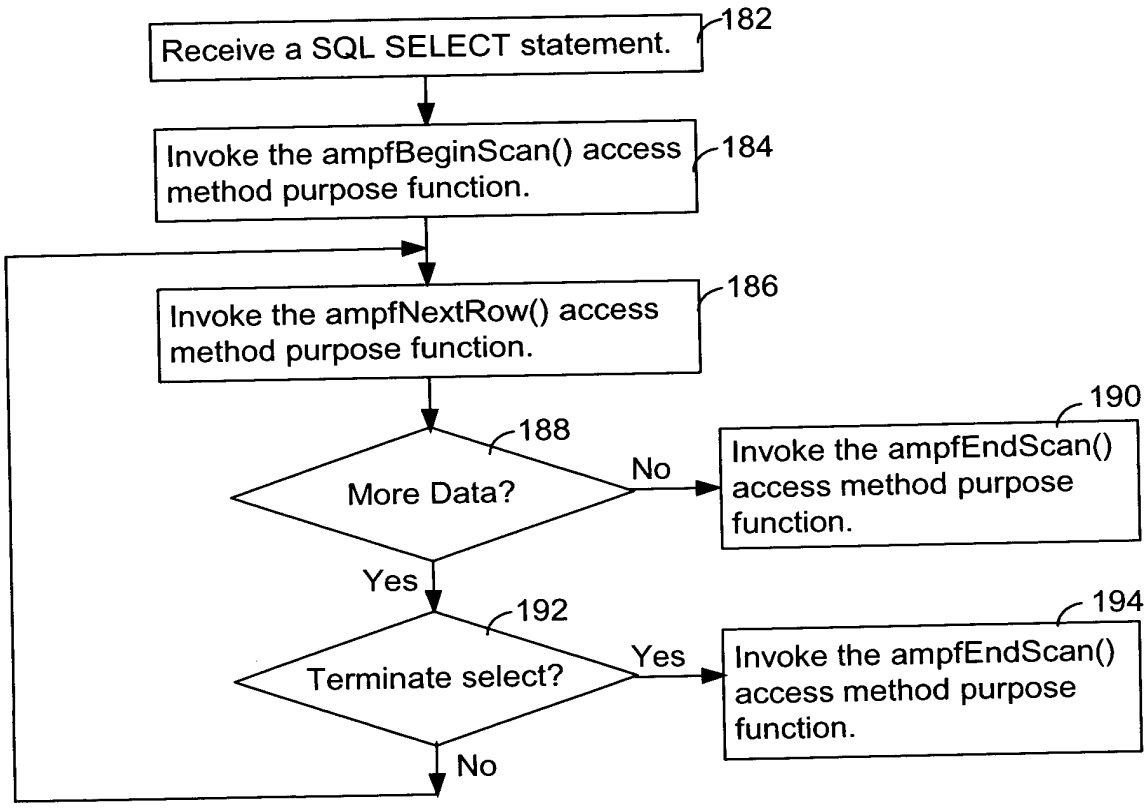

Fig. 11

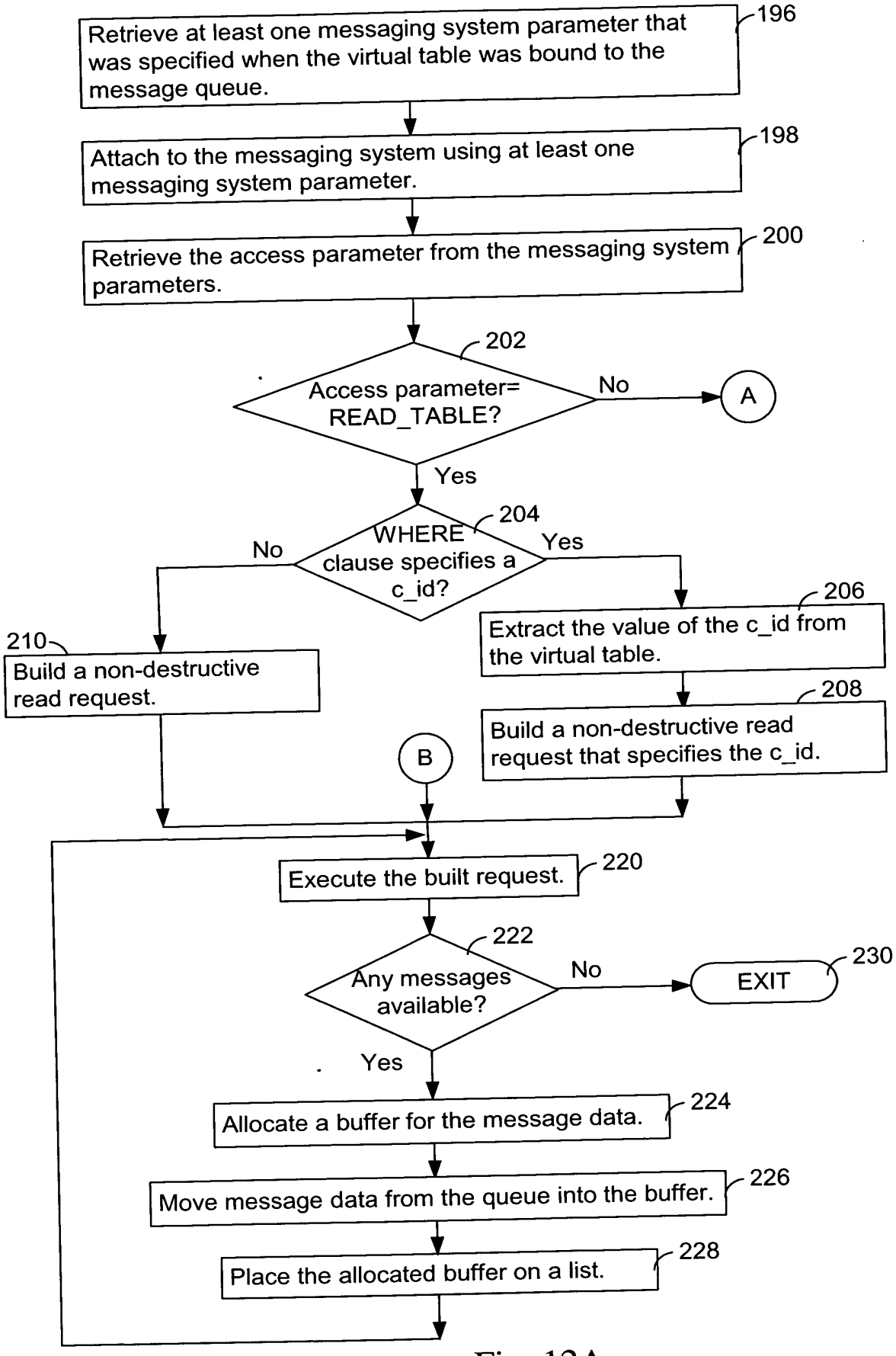
Retrieve at least one messaging system parameter that was specified when the virtual table was bound to the message queue.
196
Attach to the messaging system using at least one messaging system parameter.
198
Retrieve the access parameter from the messaging system parameters.
200
202
Access parameter= READ_TABLE?
No
A
Yes
204
WHERE clause specifies a c_id?
No
Yes
210
Build a non-destructive read request.
206
Extract the value of the c_id from the virtual table.
208
Build a non-destructive read request that specifies the c_id.
B
Execute the built request.
220
222
Any messages available?
No
EXIT
230
Yes
Allocate a buffer for the message data.
224
Move message data from the queue into the buffer.
226
Place the allocated buffer on a list.
228

Fig. 12A

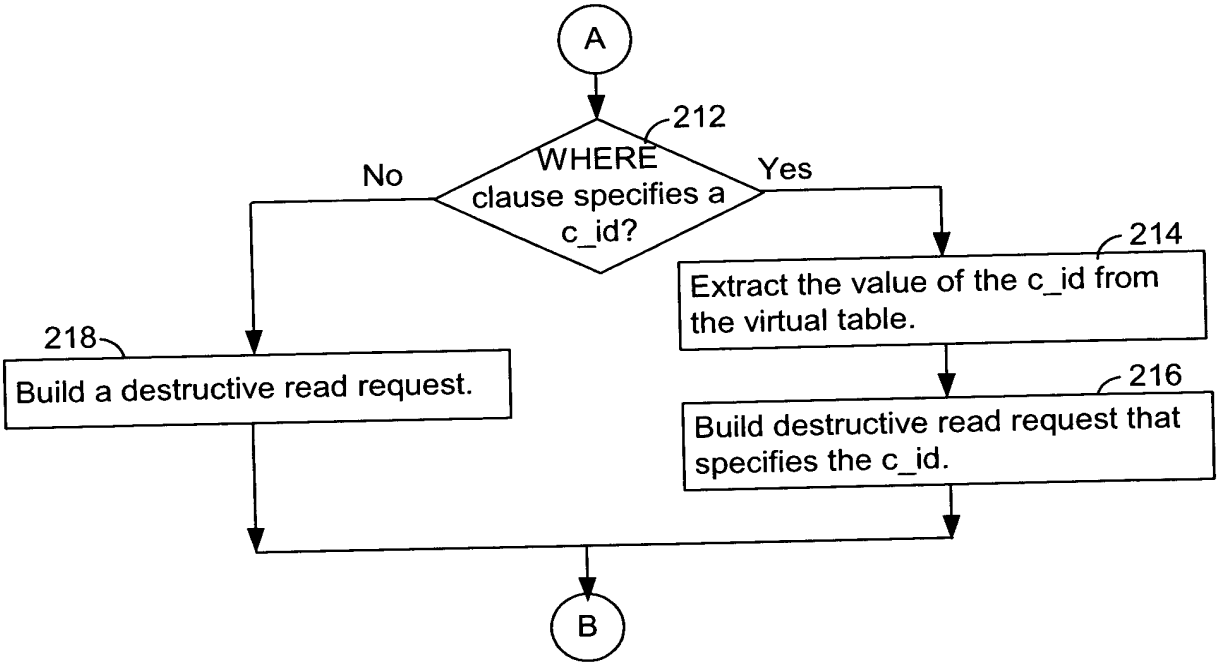

Fig. 12B

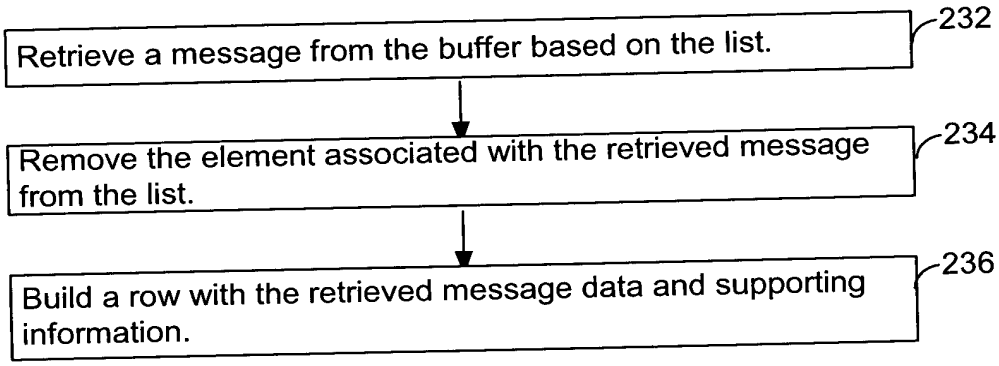
Retrieve a message from the buffer based on the list.
232
Remove the element associated with the retrieved message from the list.
234
Build a row with the retrieved message data and supporting information.
236

Fig. 13

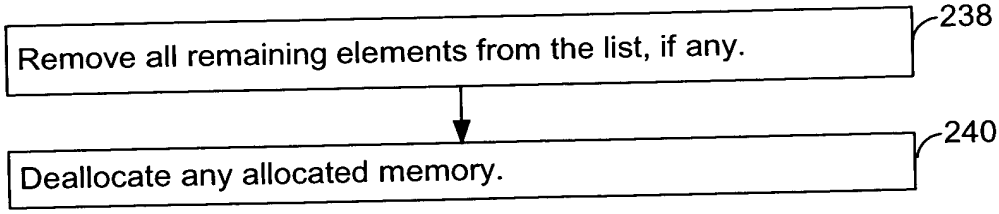

Fig. 14

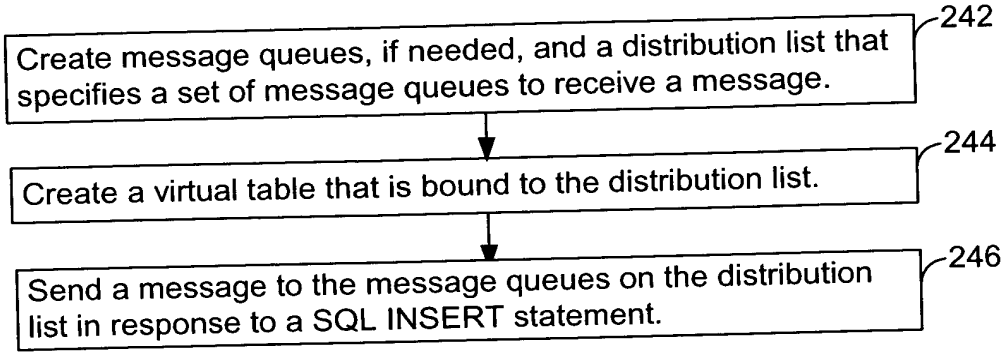
Create message queues, if needed, and a distribution list that specifies a set of message queues to receive a message.
242
Create a virtual table that is bound to the distribution list.
244
Send a message to the message queues on the distribution list in response to a SQL INSERT statement.
246

Fig. 15

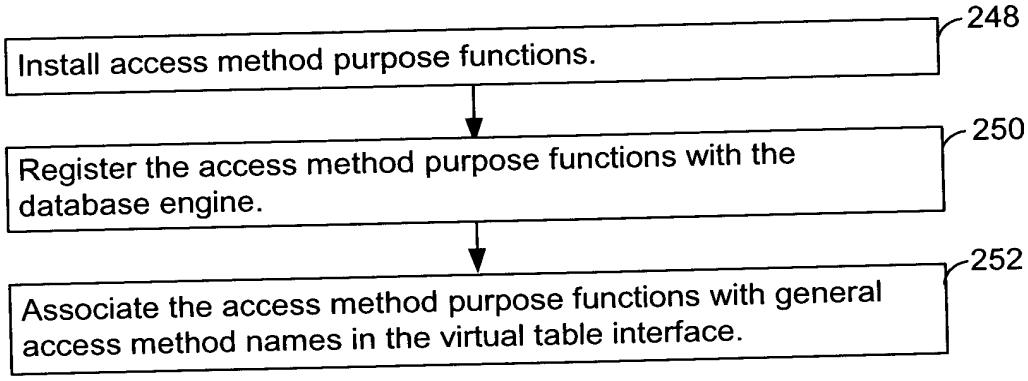
Install access method purpose functions.
248
Register the access method purpose functions with the database engine.
250
Associate the access method purpose functions with general access method names in the virtual table interface.
252

Fig. 16A

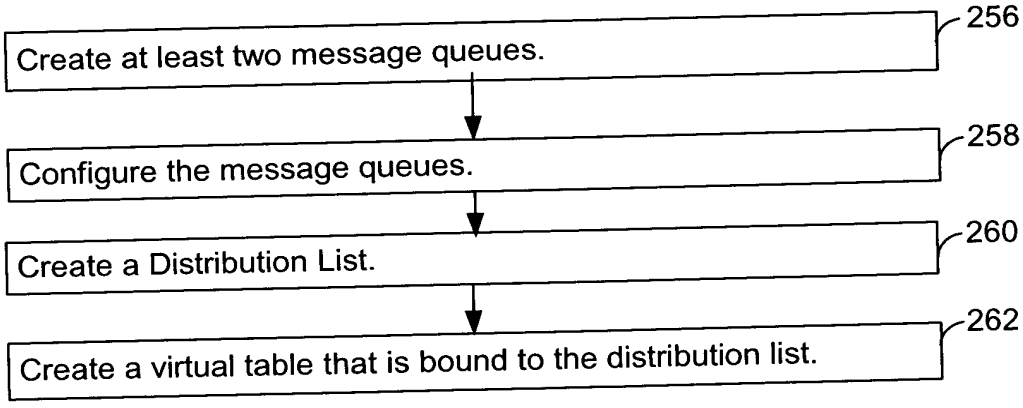

Fig. 16B

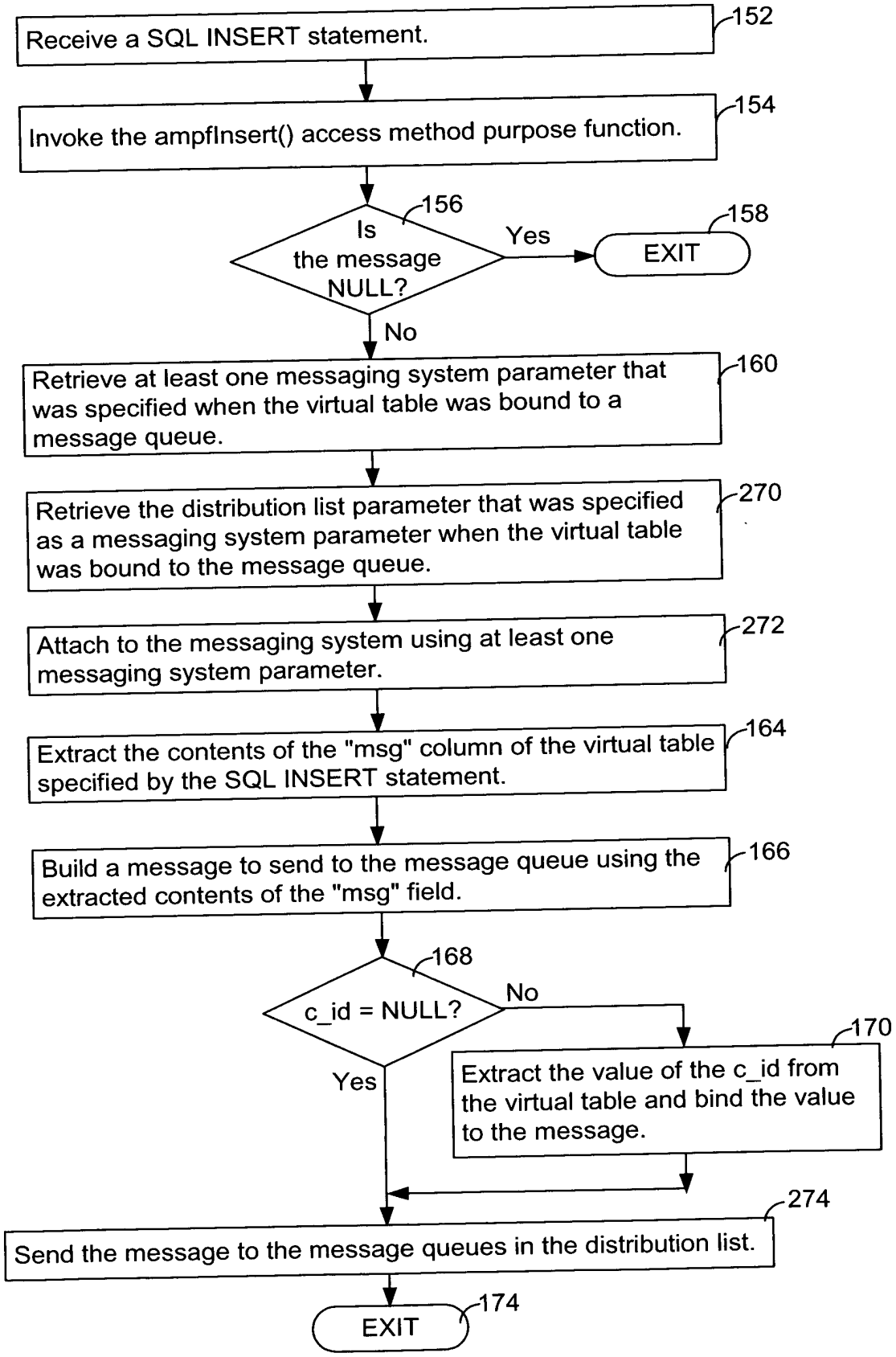
Receive a SQL INSERT statement.
152
Invoke the ampfInsert() access method purpose function.
154
Is the message NULL?
156
Yes
EXIT
158
No
Retrieve at least one messaging system parameter that was specified when the virtual table was bound to a message queue.
160
Retrieve the distribution list parameter that was specified as a messaging system parameter when the virtual table was bound to the message queue.
270
Attach to the messaging system using at least one messaging system parameter.
272
Extract the contents of the "msg" column of the virtual table specified by the SQL INSERT statement.
164
Build a message to send to the message queue using the extracted contents of the "msg" field.
166
c_id = NULL?
168
No
Yes
Extract the value of the c_id from the virtual table and bind the value to the message.
170
Send the message to the message queues in the distribution list.
274
EXIT
174

FIG. 17